(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09237283 A**

(43) Date of publication of application: **09.09.97**

(51) Int. Cl **G06F 17/50**

(21) Application number: **08043204**

(22) Date of filing: **29.02.96**

(71) Applicant: **RICOH CO LTD**

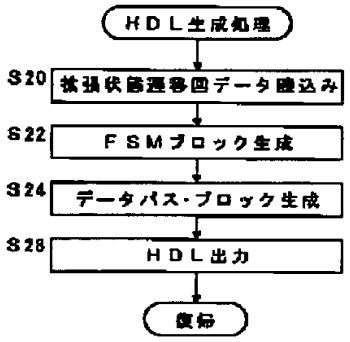
(72) Inventor: **YAMADA TAKAMITSU**
SUGAYA KAZUNOBU
OKA ZENJI

(54) LSI FUNCTION DESIGN SUPPORT DEVICE

(57) Abstract:

PROBLEM TO BE SOLVED: To generate hardware description data which corresponds to various targets and which can be logically synthesized with high quality from extension state transition drawing data.

SOLUTION: Hardware description data by HDL(hardware description language) is generated from extension state transition drawing data by permitting CPU to execute following operation based on a program stored in a memory. Description data by HDL of a part operating as a finite state machine is generated (S22) based on inputted extension state transition drawing data. Then, description data by HDL of a data path part is generated (S24). Data are outputted as hardware description data in a reregister transfer level being a design object (S28). Thus, hardware description data of LSI, in which an FSM part and the data path part are constituted as different blocks, can be obtained. Consequently, the logic synthesis tool of a high compression rate can be used on the FSM (finite state machine block) part and debugging after logic synthesis becomes easy.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－237283

(43)公開日　平成９年(1997)９月９日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ		技術表示箇所
Ｇ０６Ｆ　17/50			Ｇ０６Ｆ　15/60	６５４Ａ	

審査請求　未請求　請求項の数７　ＯＬ　（全 16 頁）

(21)出願番号　特願平8－43204

(22)出願日　平成８年(1996)２月29日

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号

(72)発明者　山田　孝光
東京都大田区中馬込１丁目３番６号　株式会社リコー内

(72)発明者　菅谷　和伸
東京都大田区中馬込１丁目３番６号　株式会社リコー内

(72)発明者　岡　善治
東京都大田区中馬込１丁目３番６号　株式会社リコー内

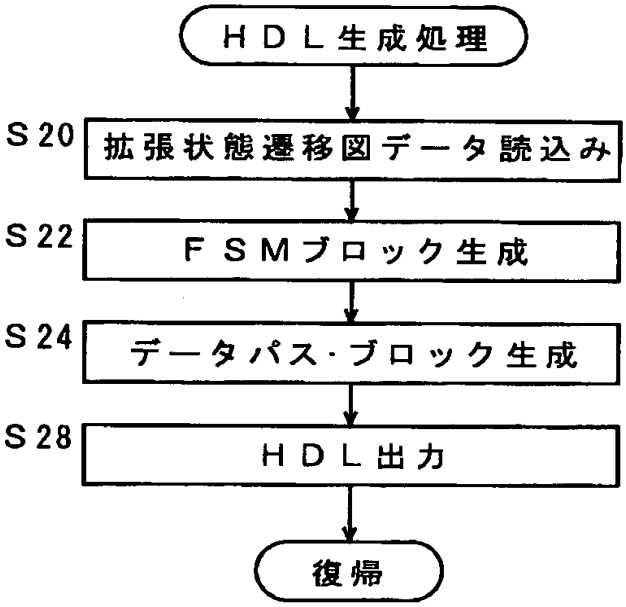
(74)代理人　弁理士　青山　葆　（外１名）

(54)【発明の名称】　ＬＳＩ機能設計支援装置

(57)【要約】

【課題】　種々の目的に応じた質の高い論理合成が可能となるハードウェア記述データを拡張状態遷移図データから生成する。

【解決手段】　メモリに格納されたプログラムに基づきＣＰＵが以下のように動作することにより、拡張状態遷移図データからＨＤＬによるハードウェア記述データを生成する。入力された拡張状態遷移図データに基づき、まず有限状態機械として動作する部分のＨＤＬによる記述データを生成し（Ｓ２２）、次にデータパス部分のＨＤＬによる記述データを生成し（Ｓ２４）、これらのデータを設計対象のレジスタ転送レベルのハードウェア記述データとして出力する（Ｓ２８）。これによりＦＳＭ部分とデータパス部分とが別のブロックとして構成されるＬＳＩのハードウェア記述データが得られるため、ＦＳＭ部分については圧縮率の高い論理合成ツールが使用でき、また論理合成後のデバッグ等が容易になる。



1

【特許請求の範囲】

【請求項１】　操作者による入力操作に基づき、任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のＬＳＩの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、

前記拡張状態遷移図データに基づき、前記ＬＳＩにおける制御論理部分に対応する有限状態機械の記述データを生成する有限状態機械記述生成手段と、

前記拡張状態遷移図データに基づき、前記ＬＳＩにおけるデータパス部分の記述データを生成するデータパス記述生成手段と、

有限状態機械記述生成手段およびデータパス記述生成手段によって生成された記述データから、制御論理部分とデータパス部分とが分離された構成の前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするＬＳＩ機能設計支援装置。

【請求項２】　請求項１に記載のＬＳＩ機能設計支援装置において、

前記データパス記述生成手段は、前記データパス部分における組合せ回路の記述データを生成する組合せ回路記述生成手段と、前記データパス部分におけるレジスタ群の記述データを生成するレジスタ記述生成手段とを有し、

前記ハードウェア記述生成手段は、データパス部分が組合せ回路とレジスタ群とに分離された構成の前記ＬＳＩの前記ハードウェア記述データを生成する、ことを特徴とするＬＳＩ機能設計支援装置。

【請求項３】　前記制御論理部分を動作させるクロックと前記データパス部分を動作させるクロックとが異なる請求項１に記載のＬＳＩ機能設計支援装置において、

前記ＬＳＩの外部からのテストモード信号およびテストクロック信号の入力が可能であって、該テストモード信号が所定値のときに、該テストクロック信号を、前記制御論理部分を動作させるクロック信号および前記データパス部分を動作させるクロック信号として供給するテスト回路の記述データを生成するテスト回路記述生成手段を備え、

前記ハードウェア記述生成手段は、前記テスト回路記述生成手段によって生成された記述データを用いて、前記テスト回路が挿入された前記ＬＳＩの前記ハードウェア記述データを生成する、ことを特徴とするＬＳＩ機能設計支援装置。

【請求項４】　操作者による入力操作に基づき、状態遷移図により設計対象のＬＳＩの動作を記述した状態遷移図データを生成する入力データ生成手段と、

前記状態遷移図データによって表される前記ＬＳＩの各状態を識別する状態コードを生成する状態コード生成手段と、

2

前記ＬＳＩの製造技術に対応する論理素子である複数ビット分のフリップフロップの機能を有するマクロが登録されているライブラリを記憶する記憶手段と、

前記状態コードを表現するために必要なビット数分のフリップフロップの機能を有するマクロを前記ライブラリから選択するマクロ選択手段と、

マクロ選択手段によって選択されたマクロを前記状態コードの格納部とする前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするＬＳＩ機能設計支援装置。

【請求項５】　操作者による入力操作に基づき、状態遷移の階層的記述が可能である拡張状態遷移図により設計対象のＬＳＩの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、

前記拡張状態遷移図データによって表される前記ＬＳＩの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、

前記ＬＳＩが前記マクロ状態にあるときに値が「１」となり、前記マクロ状態ではないときに値が「０」となるフラグ信号の記述データを生成するフラグ記述生成手段と、

前記マクロ状態における下位の状態の遷移クロックを前記フラグ信号が「１」のときに供給させ、前記フラグ信号が「０」のときに該遷移クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、

フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記遷移クロックの供給と停止が制御される前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするＬＳＩ機能設計支援装置。

【請求項６】　操作者による入力操作に基づき、状態遷移の階層的記述および任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のＬＳＩの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、

前記拡張状態遷移図データによって表される前記ＬＳＩの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、

前記ＬＳＩが前記マクロ状態にあるときに値が「１」となり、前記マクロ状態ではないときに値が「０」となるフラグ信号の記述データを生成するフラグ記述生成手段と、

前記マクロ状態の下位の状態に割り付けられたレジスタ動作のためのクロックを前記フラグ信号が「１」のときに供給させ、前記フラグ信号が「０」のときに該クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、

フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記クロックの供給と停止が制御される前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするＬＳＩ機能設計支援装置。

【請求項７】　操作者による入力操作に基づき、状態遷移の階層的記述および任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のＬＳＩの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、

前記拡張状態遷移図データによって表される前記ＬＳＩの各状態を識別する状態コードを生成する状態コード生成手段と、

前記ＬＳＩの製造技術に対応する論理素子である複数ビット分のフリップフロップの機能を有するマクロが登録されているライブラリを記憶する記憶手段と、

前記状態コードを表現するために必要なビット数分のフリップフロップの機能を有するマクロを前記ライブラリから選択するマクロ選択手段と、

前記拡張状態遷移図データによって表される前記ＬＳＩの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、

前記ＬＳＩが前記マクロ状態にあるときに値が「１」となり、前記マクロ状態ではないときに値が「０」となるフラグ信号の記述データを生成するフラグ記述生成手段と、

前記マクロ状態における下位の状態の遷移クロックを前記フラグ信号が「１」のときに供給させ、前記フラグ信号が「０」のときに該遷移クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、

前記拡張状態遷移図データおよびマクロ選択手段によって選択されたマクロに基づき、フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記ＬＳＩにおける制御論理部分である有限状態機械であってマクロ選択手段により選択されたマクロを前記状態コードの格納部とし前記フラグ信号により前記遷移クロックの供給と停止が制御される有限状態機械の記述データを生成する有限状態機械記述生成手段と、

前記拡張状態遷移図データに基づき、前記ＬＳＩにおけるデータパス部分の記述データを生成するデータパス記述生成手段と、

有限状態機械記述生成手段およびデータパス記述生成手段によって生成された記述データから、制御論理部分とデータパス部分とが分離された構成の前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするＬＳＩ機能設計支援装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ＬＳＩ（大規模集積回路）の機能設計を支援する装置に関し、更に詳しくは、ＬＳＩの動作を状態遷移図形式で記述したデータからレジスタ転送レベルのハードウェア記述言語によるハードウェア記述データを生成する機能を備えたＬＳＩ機能設計支援装置に関する。

【０００２】

【従来の技術】近年、ＬＳＩの開発においてハードウェア記述言語(Hardware Description Language)（以下「ＨＤＬ」という）を用いたトップダウン設計手法がとられるようになってきている。すなわち、機能設計の段階において設計対象のＬＳＩをレジスタ転送レベルのＨＤＬで記述したハードウェア記述データが作成され、このハードウェア記述データから論理合成ツールによってゲートレベルの回路データが生成されるようになっている。

【０００３】このようなトップダウン設計手法において、レジスタ転送レベルのＨＤＬによるハードウェア記述データの作成を容易にするために、設計者が状態遷移図を基本とする形式でグラフィック端末より設計対象のＬＳＩの動作を示すデータ（設計情報）を入力するためのＬＳＩ機能設計支援装置が提案されている。この装置では、入力された状態遷移図形式のデータからレジスタ転送レベルのＨＤＬによるハードウェア記述データが自動的に生成される。例えば、特開平３－４１５６７号公報には、状態遷移図および機能図をＣＡＤ(Computer Aided Design)データとして対話形式でデータベース登録し、このデータベースより対応するハードウェア記述言語に変換してファイル出力する状態遷移図設計システムが開示されている。これによれば、デジタル回路の設計システムにおけるユーザインタフェイスが向上し、設計効率を向上させることができるという効果が得られる。

【０００４】上記のＬＳＩ機能設計支援装置によってハードウェア記述データを生成する場合、その記述スタイルによって、論理合成後の回路の品質や性能が異なる。このため、目的に応じた質の高い論理合成を行えるようなハードウェア記述が望まれる。

【０００５】例えば、論理合成後の回路規模の縮小化を重視する場合には、圧縮率の高い論理圧縮アルゴリズムによる論理合成ツールを使用できるようなハードウェア記述が望まれる。また、製造後の回路のデバッグやテストに手間を要すると予想される場合には、デバッグやテストが容易となるようなハードウェア記述が望まれる。ところで前述の状態遷移図によって表される各状態を示す状態コードを格納するフリップフロップは、通常、複数個から成り、同一クロックに同期する。したがって、クロックスキューが生じない安定した回路を実現するためには、状態コードを格納する複数のフリップフロップ

が一つの回路ブロックとして実現されるようなハードウェア記述が望まれる。また、近年の携帯機器の普及によりＬＳＩの消費電力の低減が重要になっており、携帯機器に使用されるＬＳＩについては、動作速度よりも消費電力を抑えることが優先され、これに対応したハードウェア記述が望まれる。

【０００６】

【発明が解決しようとする課題】しかし、従来のＬＳＩ機能設計支援装置では、ハードウェア記述データの生成において上記の点は考慮されていなかった。すなわち、設計者によって入力された状態遷移図形式のデータ（設計情報）により設計対象のＬＳＩの動作が有限状態機械（以下「ＦＳＭ」という）と所定状態におけるレジスタ動作の記述（データパスの記述）として表現されている場合には、このデータから従来のＬＳＩ設計支援装置によって生成されるハードウェア記述データに対しては、圧縮率の高いＦＳＭ専用の論理圧縮アルゴリズムに基づくツールを使用することができなかった。

【０００７】また、従来のＬＳＩ機能設計支援装置には、状態遷移図形式の入力データからレジスタ転送レベルのハードウェア記述データを生成する際に、論理合成後の回路のデバッグやタイミング解析の容易化、消費電力の低減などについて考慮しているものはなかった。さらに、同一のクロックに同期する複数のフリップフロップについても、論理合成後にそれらが一つの回路ブロックとして実現されるようなハードウェア記述データを生成するＬＳＩ機能設計支援装置は存在しなかった。

【０００８】そこで本発明では、回路規模（チップ面積）の縮小化や、デバッグ等の容易化、消費電力の低減、回路動作の安定化等の各種の目的に応じた質の高い論理合成が可能となるようなハードウェア記述データを状態遷移図形式の入力データから生成することができるＬＳＩ機能設計支援装置を提供することを目的とする。

【０００９】

【課題を解決するための手段】上記課題を解決するためになされた本発明に係る第１のＬＳＩ機能設計支援装置では、操作者による入力操作に基づき、任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のＬＳＩの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データに基づき、前記ＬＳＩにおける制御論理部分に対応する有限状態機械の記述データを生成する有限状態機械記述生成手段と、前記拡張状態遷移図データに基づき、前記ＬＳＩにおけるデータパス部分の記述データを生成するデータパス記述生成手段と、有限状態機械記述生成手段およびデータパス記述生成手段によって生成された記述データから、制御論理部分とデータパス部分とが分離された構成の前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴としている。

【００１０】本発明に係る第２のＬＳＩ機能設計支援装置では、上記第１のＬＳＩ機能設計支援装置において、前記データパス記述生成手段は、前記データパス部分における組合せ回路の記述データを生成する組合せ回路記述生成手段と、前記データパス部分におけるレジスタ群の記述データを生成するレジスタ記述生成手段とを有し、前記ハードウェア記述生成手段は、データパス部分が組合せ回路とレジスタ群とに分離された構成の前記ＬＳＩの前記ハードウェア記述データを生成する、ことを特徴としている。

【００１１】本発明に係る第３のＬＳＩ機能設計支援装置では、前記制御論理部分を動作させるクロックと前記データパス部分を動作させるクロックとが異なる上記第１のＬＳＩ機能設計支援装置において、前記ＬＳＩの外部からのテストモード信号およびテストクロック信号の入力が可能であって、該テストモード信号が所定値のときに、該テストクロック信号を、前記制御論理部分を動作させるクロック信号および前記データパス部分を動作させるクロック信号として供給するテスト回路の記述データを生成するテスト回路記述生成手段を備え、前記ハードウェア記述生成手段は、前記テスト回路記述生成手段によって生成された記述データを用いて、前記テスト回路が挿入された前記ＬＳＩの前記ハードウェア記述データを生成する、ことを特徴としている。

【００１２】本発明に係る第４のＬＳＩ機能設計支援装置では、操作者による入力操作に基づき、状態遷移図により設計対象のＬＳＩの動作を記述した状態遷移図データを生成する入力データ生成手段と、前記状態遷移図データによって表される前記ＬＳＩの各状態を識別する状態コードを生成する状態コード生成手段と、前記ＬＳＩの製造技術に対応する論理素子である複数ビット分のフリップフロップの機能を有するマクロが登録されているライブラリを記憶する記憶手段と、前記状態コードを表現するために必要なビット数分のフリップフロップの機能を有するマクロを前記ライブラリから選択するマクロ選択手段と、マクロ選択手段によって選択されたマクロを前記状態コードの格納部とする前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴としている。

【００１３】本発明に係る第５のＬＳＩ機能設計支援装置では、操作者による入力操作に基づき、状態遷移の階層的記述が可能である拡張状態遷移図により設計対象のＬＳＩの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データによって表される前記ＬＳＩの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、前記ＬＳＩが前記マクロ状態にあるときに値が「１」となり、前記マクロ状態ではないときに値が

「０」となるフラグ信号の記述データを生成するフラグ記述生成手段と、前記マクロ状態における下位の状態の遷移クロックを前記フラグ信号が「１」のときに供給させ、前記フラグ信号が「０」のときに該遷移クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記遷移クロックの供給と停止が制御される前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴としている。

【００１４】本発明に係る第６のＬＳＩ機能設計支援装置では、上記第５のＬＳＩ機能設計支援装置において、上記ゲート記述生成手段に代えて、前記マクロ状態の下位の状態に割り付けられたレジスタ動作のためのクロックを前記フラグ信号が「１」のときに供給させ、前記フラグ信号が「０」のときに該クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段を備え、上記ハードウェア記述生成手段は、上記フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記クロックの供給と停止が制御される前記ＬＳＩをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成することを特徴としている。

【００１５】本発明に係る第７のＬＳＩ機能設計支援装置は、上記第１、第４、および第５のＬＳＩ機能設計支援装置における上記特徴を全て備えた構成としている。

【００１６】

【発明の効果】本発明に係る第１のＬＳＩ機能設計支援装置によれば、制御論理部分（ＦＳＭの部分）とデータパス部分とが分離された構成のＬＳＩのハードウェア記述データが生成されるため、ＦＳＭの部分の論理合成については、圧縮率の高いＦＳＭ専用の圧縮アルゴリズムによる論理合成ツールが使用可能となり、回路規模の縮小化を図ることができる。また、論理合成後の回路のデバッグやタイミング解析を、ＦＳＭとデータパス部分に分けて実施することができるため、デバッグやタイミング解析が容易となり、ＬＳＩ設計の作業効率が向上する。

【００１７】本発明に係る第２のＬＳＩ機能設計支援装置によれば、データパス部分が更に組合せ回路とレジスタ群とに分離された構成のＬＳＩのハードウェア記述データが生成されるため、論理合成を組合せ回路部分とレジスタ群に分けて行うことができ、また、論理合成後の回路のデバッグやタイミング解析も組合せ回路部分とレジスタ群に分けて実施することができる。これにより、回路規模が更に縮小され、デバッグやタイミング解析が更に容易になる。

【００１８】本発明に係る第３のＬＳＩ機能設計支援装置によれば、ＬＳＩ外部から入力されるテストモード信号によりテストクロック信号のＦＳＭの部分およびデータパス部分への供給を可能とするテスト回路が挿入された構成のＬＳＩのハードウェア記述データが生成される。これにより、機能設計の段階からテスト容易化設計が実施されることになり、論理合成後のテスト回路挿入のための作業が不要となるため、ＬＳＩ設計の作業効率が更に向上する。

【００１９】本発明に係る第４のＬＳＩ機能設計支援装置によれば、複数ビット分のフリップフロップの機能を有するマクロを状態コードの格納部とするＬＳＩのハードウェア記述が生成され、これに対して論理合成を行うことにより、ＬＳＩにおいて状態コードを格納する部分が一つの回路ブロック（状態レジスタ）として実現される。これにより、クロックスキューの発生が防止され、ＬＳＩは安定に動作する。

【００２０】本発明に係る第５のＬＳＩ機能設計支援装置によれば、生成されるハードウェア記述データに基づいて得られるＬＳＩは、その状態がマクロ状態か否かを示すフラグ信号によりそのマクロ状態の下位の状態の遷移クロックの供給と停止が制御される構成となる。これにより、マクロ状態の下位の状態コードを格納する状態レジスタには、その下位の状態遷移が生じない期間中（そのマクロ状態以外の状態となる期間中）は遷移クロックの供給が停止されるため、ＬＳＩの消費電力が低減される。

【００２１】本発明に係る第６のＬＳＩ機能設計支援装置によれば、生成されるハードウェア記述データに基づいて得られるＬＳＩは、その状態がマクロ状態か否かを示すフラグ信号により、そのマクロ状態の下位の状態に割り付けられたレジスタ動作のためのクロックの供給と停止が制御される構成となる。これにより、マクロ状態の下位の状態に割り付けられたレジスタ動作に対しては、その下位の状態遷移が生じない期間中（そのマクロ状態以外の状態となる期間中）はクロックの供給が停止されるため、ＬＳＩの消費電力が低減される。

【００２２】本発明に係る第７のＬＳＩ機能設計支援装置は、上記第１、第４、および第５のＬＳＩ機能設計支援装置の特徴を全て備えている。したがって、第７のＬＳＩ機能設計支援装置によれば、論理合成の効率化による回路規模の縮小、論理合成語の回路のデバッグやタイミング解析の容易化、状態レジスタにおけるクロックスキューの発生防止による動作の安定化、および消費電力の低減を図ることができる。

【００２３】

【発明の実施の形態】

＜実施形態１＞以下、本発明の一実施形態（以下「実施形態１」という）であるＬＳＩ機能設計支援装置について説明する。このＬＳＩ機能設計支援装置は、汎用コンピュータとそれによって実行されるコンピュータ･プロ

グラムとによって実現される。

【００２４】図１は、本実施形態のＬＳＩ機能設計支援装置のハードウェアであるコンピュータの構成を示す概略ブロック図である。本ＬＳＩ機能設計支援装置のハードウェアは、ＣＰＵ１６とメモリ１８などから成る本体１０と、後述の拡張遷移図データやライブラリデータなどを格納するハードディスク装置１２と、キーボードやマウス等の入力装置１４と、ＣＲＴディスプレイ等の表示装置２０とから構成されている。このような構成において、本体１０のメモリ１８に所定のプログラムが格納され、ＣＰＵ１６がそのプログラムに基づいて動作することにより、ＬＳＩの機能設計を支援するための種々の機能が実現される。

【００２５】図２は、上記ＬＳＩ機能設計支援装置における処理を示すフローチャートである。上記ＬＳＩ機能設計支援装置による設計では、まずステップＳ１０において、設計者である操作者が、表示装置２０を見ながらキーボードやマウス等の入力装置１４を操作することにより、設計対象のＬＳＩであるデジタル回路の動作を示す情報を入力する。本ＬＳＩ機能設計支援装置では、状態遷移図の形式で設計対象のＬＳＩの動作を示す情報を入力することができ、このとき、その状態遷移図で示される任意の状態における演算処理等のレジスタ動作を記述できるようになっている。具体的には、状態遷移図で示される状態におけるデータパス部分の動作を示す代入文や演算式などをＨＤＬで記述する。以下、このようなレジスタ動作の記述機能の追加等により拡張された状態遷移図を「拡張状態遷移図」という（後述の拡張状態遷移図では、状態遷移の階層的記述の機能も追加されている）。このようなＬＳＩの設計情報入力のための拡張状態遷移図としては、例えば「オープンチャート」を用いることができる（山田孝光、安井隆、岡善治、「機能エントリ･ツールを用いたソーティング回路の設計」、ＣＱ出版社、雑誌〝インターフェース〟、1995年３月号, pp. 160-166　参照）。

【００２６】図４は、本ＬＳＩ機能設計支援装置により入力された設計情報である拡張状態遷移図の一例を設計対象のＬＳＩの入出力の定義とともに示す図である。図４には、状態Ｓ１を示す円の横に「Ｃ＝Ａ＋Ｂ」と記載されており、これは、状態Ｓ１において実行される演算すなわちレジスタ動作を示している。この拡張状態遷移図によって示される設計情報が操作者により入力されると、図５に示すようなデータ構造の設計データがメモリ１８内に生成され、これがハードディスク装置１２に拡張状態遷移図データとして格納される。

【００２７】このようにして設計情報が入力されて拡張状態遷移図データが生成されると、ステップＳ１２において、この拡張状態遷移図データ対し所定のチェックを行う。すなわち、状態の遷移先が存在するか否か等の状態遷移図に関するチェック（状態遷移図チェック）や、レジスタ動作を記述したハードウェア記述の文法チェック（ＨＤＬ文法チェック）、論理合成の可能性のチェック（合成チェック）を行う。

【００２８】上記各種のチェックの結果、エラーが発見されればステップＳ１０へ戻り（ステップＳ１４）、エラーがなくなるまでステップＳ１０～Ｓ１４が繰り返し実行される。そしてエラーがなくなればステップＳ１６へ進み、操作者が所定の操作によりＨＤＬの生成方式を指定する。すなわち、ハードウェア記述データの生成に際して、論理合成の効率（ＬＳＩのチップ面積の縮小化）を優先するか、消費電力の低減を優先するか、デバッグの容易性を優先するか等の優先事項を指定する。

【００２９】ＨＤＬ生成方式が指定されると、ステップＳ１８において、拡張状態遷移図データからハードウェア記述データを生成し（以下、このハードウェア記述データを生成する処理を「ＨＤＬ生成処理」という）、ＬＳＩ機能設計支援装置の処理を終了する。このようにして得られたハードウェア記述データは、設計対象のＬＳＩをレジスタ転送レベルで記述したものであり、このハードウェア記述データから、既存の論理合成ツールにより、そのＬＳＩの製造技術に対応するテクノロジ･ライブラリのマクロを用いて論理回路データが生成される。

【００３０】図３は、図２のステップＳ１８におけるＨＤＬ生成処理を実行するサブルーチンを示すフローチャートである。本ＬＳＩ機能設計支援装置は、このフローチャートに示す処理により、上記の拡張状態遷移図データからレジスタ転送レベルのハードウェア記述データを生成する。以下、このＨＤＬ生成処理について図３を参照しつつ説明する。なお本実施形態では、ハードウェア記述言語（ＨＤＬ）としてVerilog-ＨＤＬを使用する。

【００３１】ＨＤＬ生成処理では、まずステップＳ２０において、ハードディスク装置１２に格納された拡張状態遷移図データをメモリ１８へ読み込む。次にステップＳ２２において、読み込まれた拡張状態遷移図データによって示されるＦＳＭへの入力信号を入力ポートに持ち、ＦＳＭからの出力信号と現在の状態を示す信号（以下「状態信号」という）とを出力ポートに持つような一つの階層ブロックを設定し、これをＦＳＭとして動作する階層ブロック（以下「有限状態機械ブロック」という）として記述したハードウェア記述データを生成する。例えば図４に示した拡張状態遷移図データに対しては、有限状態機械ブロックのハードウェア記述データとして図８に示すようなデータを生成する。その後、ステップＳ２４において、拡張状態遷移図データによって示されるデータパス部分（図４に示した例では「Ｃ＝Ａ＋Ｂ」という記述に相当する部分）の入力パスと、前記ＦＳＭの階層ブロックの出力信号である状態信号とを入力ポートに持ち、データパス部分の出力パスを出力ポートに持つような一つの階層ブロックを設定し、これをデータパス部分として動作する階層ブロック（以下「データ

パス･ブロック」という）として記述したハードウェア記述データを生成する。例えば図４に示した拡張状態遷移図データに対しては、データパス･ブロックのハードウェア記述データとして図９に示すようなデータを生成する。このようにして、有限状態機械ブロックとデータパス･ブロックの両階層ブロックのハードウェア記述データを出力した後は、ステップＳ２８において、図８に示す有限状態機械ブロックFSMの記述データおよび図９に示すデータパス･ブロックREGACTの記述データから成るハードウェア記述データを、これらの階層ブロックから構成される最上位の階層ブロックＴＯＰのハードウェア記述データとして、すなわち図１０に示すブロック図に相当するハードウェア記述データとして出力する。これによりＨＤＬ生成処理を終了し、このサブルーチンから復帰する。

【００３２】以上のようにして本実施形態において生成されるハードウェア記述データと比較するために、従来の手法により図４に示した拡張状態遷移図データから生成されたハードウェア記述データを図６に示す。このときの設計対象のＬＳＩの外部ポートの入出力関係は、図７に示すようになる。

【００３３】本実施形態において図４に示した拡張状態遷移図データから生成されるハードウェア記述データは、前述のように図８および図９に示す通りであり、ＦＳＭに相当する部分とデータパスに相当する部分とが別個の階層ブロックとして構成されるようなハードウェア記述データとなっている。したがって、ＦＳＭ専用の論理圧縮アルゴリズムによる論理合成ツールを、生成されたハードウェア記述のうちのＦＳＭに相当する部分に適用することが可能となり、これにより高い圧縮率が得られる。また、論理合成後の回路デバッグやタイミング解析はＦＳＭとデータパス部分を分離して実施することができるため、デバッグやタイミング解析が容易となり、設計の作業効率が向上する。

【００３４】＜実施形態２＞上記実施形態において生成されるハードウェア記述データによれば、設計対象のＬＳＩは有限状態機械ブロックとデータパス･ブロックとが分離された構成となるが、データパス･ブロックを、更に、組合せ回路のみから成る階層ブロックとレジスタ群からなる階層ブロックとに分離した構成となるようにすれば、論理合成やデバッグをこれらの階層ブロック毎に行うことができるため、論理合成の効率が更に向上し、論理合成後の回路のデバッグやタイミング解析が更に容易となる。このためには、図３のステップＳ２４におけるデータパス･ブロックのハードウェア記述データの生成を以下のように変更すればよい（以下、このように変更した本発明の実施の形態を「実施形態２」という）。

【００３５】すなわち、図１１のフローチャートに示すように、まずステップＳ１０２において、ステップＳ２０で読み込まれた拡張状態遷移図データによって示されるデータパス部分からクロック信号によって引き起こされるデータロード動作を削除したものを抽出し、これを一つの階層ブロック（以下「組合せ回路ブロック」という）として記述したハードウェア記述データを生成する。例えば図４に示した拡張状態遷移図データに対しては、組合せ回路ブロックのハードウェア記述データとして図１２に示すようなデータを生成する。次にステップＳ１０４において、前記データパス部分のうち内部レジスタ（レジスタ群）を構成するフリップフロップへと論理合成によって変換されるものを抽出し、これを一つの階層ブロック（以下「レジスタ･ブロック」という）として記述したハードウェア記述データを生成する。例えば図４に示した拡張状態遷移図データに対しては、レジスタ･ブロックのハードウェア記述データとして図１３に示すようなデータを生成する。このようにして得られた図１２の組合せ回路ブロックCOMBの記述データと図１３のレジスタ･ブロックREGの記述データから成るハードウェア記述データによれば、図１４に示すように、データパス･ブロックREGACTにおいて組合せ回路ブロックCOMBとレジスタ･ブロックREGとが別個の階層ブロックとして構成されることになる。

【００３６】＜実施形態３＞次に、テスト容易化設計に対応した本発明の実施の形態（以下「実施形態３」という）を説明する。本実施形態では、ＨＤＬ生成処理は、図１５のフローチャートに示すようになる。このフローチャートは、上記実施形態１における図３のフローチャートにテスト回路挿入のためのステップＳ２６を追加したものである。その他の構成は、上記実施形態１と同様である。

【００３７】本実施形態では、拡張状態遷移図データからレジスタ転送レベルのハードウェア記述データを生成する際に、上記実施形態１と同様に図１５のステップＳ２０～Ｓ２４の処理を行った後、ステップＳ２６においてテスト回路挿入のための処理を行う。すなわち、外部から供給されるテストモード信号とテストクロック信号に対するハードウェア記述データとともに、テストモード信号が「０」のときに有限状態機械ブロックとデータパス･ブロックとにそれぞれに対応した通常のクロックを供給し、テストモード信号が「１」のときにテストクロックを有限状態機械ブロックおよびデータパス･ブロックに供給するマルチプレクサに対するハードウェア記述データを生成する。これにより、図１６に示す構成の回路に相当するレジスタ転送レベルのハードウェア記述データが得られる。図１６は、最上位の階層ブロックＴＯＰの構成を示すブロック図であって、この構成によれば、通常の動作時には、テストモード信号TEST_MODEが「０」とされて、有限状態機械ブロックFSMにはクロック信号ＣＬＫが、データパスブロックREGACTにはクロック信号ＣＫ２がそれぞれ供給されるが、テスト時には、

テストモード信号TEST_MODEが「１」とされて、有限状態機械ブロックFSMおよびデータパスブロックREGACTにテストクロック信号TEST_CLOCKが供給される。

【００３８】以上のように本実施形態によれば、レジスタ転送レベルのハードウェア記述データ生成においてテスト回路が挿入されるため、論理合成後においてテスト回路を挿入することなく、テスト容易化設計が実現される。これにより、ＬＳＩ設計の作業効率が向上する。

【００３９】＜実施形態４＞次に、本発明の第４の実施の形態（以下「実施形態４」という）であるＬＳＩ機能設計支援装置について説明する。本実施形態のＬＳＩ機能設計支援装置は、上記実施形態１～３と同一のハードウェア構成を有し（図１参照）、上記実施形態１～３と同様にして設計対象のＬＳＩの動作を記述する拡張状態遷移図データを入力する（図２のステップＳ１０～Ｓ１４参照）。しかし本実施形態では、ＨＤＬ生成処理が上記実施形態１～３と相違する。また本実施形態では、レジスタ転送レベルのハードウェア記述に使用できるマクロを集めたマクロ･ライブラリが、ハードディス装置１２に格納されている。そして、このマクロ･ライブラリには、設計対象のＬＳＩの製造技術に対応するテクノロジ･ライブラリに登録されている複数ビット分のフリップフロップの機能を有するフリップフロップ･マクロに相当するマクロが登録されている。

【００４０】以下、本実施形態のＨＤＬ生成処理について図１７に示すフローチャートを参照しつつ説明する。まずステップＳ３０において、実施形態１～３と同様にしてハードディスク装置１２内に格納された設計情報である拡張状態遷移図データを、メモリ１８に読み込む。以下では、このステップにおいて図１８に示すような状態遷移図データが読み込まれたものとして説明する。図１８において、例えば状態名「Ｓ０」はＳ０とラベル付けされた状態を示し、その横に次の状態として「Ｓ１」が記載されているのは、状態Ｓ０の次に状態Ｓ１に遷移することを示している。また、出力としての「Ｚ」の記載の有無は、「Ｚ」の記載された状態において信号名Ｚの出力信号が「１」となり、「Ｚ」の記載されていない状態において信号名Ｚの出力信号が「０」となることを示す。なお図１８に示した例では、レジスタ動作の記述は含まれていないため、メモリ１８に読み込まれるデータは、通常の状態遷移図データである。

【００４１】次のステップＳ３２では、読み込まれた状態遷移図データによって表される状態を調べ、各状態に状態コードを割り付ける（状態コード解析処理）。図１８に示した状態遷移図データの場合には、Ｓ０～Ｓ８の９個の状態が存在するため、最小ビット幅で状態コードを割り付けるとすると、４ビット幅の状態レジスタが必要となる。図１９は、図１８の状態遷移図データに対する状態コード解析の結果を示しており、この例では、状態Ｓ０～Ｓ８に４ビットの状態コード「００００」～

「１０００」がそれぞれ割り付けられている。このようにして状態コードが割り付けられると、設計対象のＬＳＩの動作を示す状態遷移図は図２２に示すようになり、これが表示装置２０に表示される。

【００４２】状態コードの割り付け後は、ステップＳ３４において、ハードディスク装置１２に格納されたマクロ･ライブラリを参照して、状態コードを格納する状態レジスタの実現に必要なフリップフロップ･マクロを求める。いま、マクロ･ライブラリには、図２０に示すように、「Ｒ４２」というマクロ名の４ビットのフリップフロップ･マクロと、「Ｒ８２」というマクロ名の８ビットのフリップフロップ･マクロとが登録されているものとする。この場合、図１９に示したように状態コードが割り付けられたとすると、フリップフロップ･マクロR42が選択され、これに対応する図２１に示すデータが、設計対象のＬＳＩのハードウェア記述に使用されるフリップフロップ･データとしてハードディスク装置１２に格納される。

【００４３】そしてステップＳ３６において、ステップＳ３０で読み込まれた状態遷移図データから、ステップ３４で得られたフリップフロップ･データを参照して、レジスタ転送レベルのハードウェア記述データを生成する。

【００４４】以上のようにして本実施形態において生成されるハードウェア記述データと比較するために、従来のＬＳＩ機能設計支援装置により図１８の状態遷移図データから生成されたハードウェア記述データを図２３に示す。図２３において、ＨＤＬの記述２０１は４ビットの状態コードの割り付けを、ＨＤＬの記述２０２は状態レジスタをそれぞれ示す。この図２３に示すハードウェア記述データから論理合成ツールによって論理回路を合成すると、図２５に示すような回路が得られる。

【００４５】これに対し、本実施形態において図１８の状態遷移図データから生成されるハードウェア記述データは、図２４に示すようになる。図２４において、ＨＤＬの記述２１１は状態コードの割り付けを、ＨＤＬの記述２１２は状態レジスタを、ＨＤＬの記述２１３は４ビットのフリップフロップ･マクロR42の使用をそれぞれ示している。この図２４に示すハードウェア記述データから論理合成ツールによって論理回路を合成すると、図２６に示すような回路が得られる。従来のＬＳＩ機能設計支援装置により得られた図２５に示した論理回路では、状態コードを格納するレジスタとして１ビットのフリップフロップＤＦＦＣＯＲが４個生成されているのに対し、図２５に示す論理回路では、状態コードを格納するレジスタとして４ビットのフリップフロップ･マクロR42が１個生成されている。

【００４６】このように本実施形態によれば、状態コードを格納する状態レジスタとして複数ビットのフリップフロップ･マクロを使用したハードウェア記述データが

得られる。このフリップフロップ･マクロは、設計対象のＬＳＩの製造技術に対応するテクノロジ･ライブラリに登録されているマクロに相当するものである。このため、上記ハードウェア記述データに対して論理合成を行うと、状態レジスタとして複数ビット分のフリップフロップが一つの回路ブロックとして生成される。これにより、クロックスキュー等が生じず安定に動作するＬＳＩを実現することができる。また、状態レジスタが一つのマクロとして実現されるため、チップ面積の点でも有利である。

【００４７】＜実施形態５＞次に、本発明の第５の実施の形態（以下「実施形態５」という）であるＬＳＩ機能設計支援装置について説明する。本実施形態のＬＳＩ機能設計支援装置も、上記実施形態１～４と同一のハードウェア構成を有し（図１参照）、上記実施形態１～４と同様にして設計対象のＬＳＩの動作を記述する拡張状態遷移図データを入力する（図２のステップＳ１０～Ｓ１４参照）。ただし、本実施形態で使用される拡張状態遷移図は、任意の状態におけるレジスタ動作を記述できるのみならず、状態遷移を階層的に記述できるものである。例えば前述のオープンチャートは、このような記述機能を有している。

【００４８】本実施形態のＨＤＬ生成処理は上記実施形態１～４と相違する。以下、本実施形態のＨＤＬ生成処理について図２７に示すフローチャートを参照しつつ説明する。まずステップＳ４０において、階層的記述が可能な拡張状態遷移図によって表現された設計情報である拡張状態遷移図データを、ハードディスク装置１２からメモリ１８に読み込む。図２８は、ステップＳ４０で読み込まれる拡張状態遷移図データの一例を示す図である。この拡張状態遷移図データによって表される状態のうち、状態ＨＳ０はその下階層として拡張状態遷移図３０４で示されるような状態遷移を持ち、状態ＨＳ１およびＨＳ２も下階層を持っている。また、状態ＨＳ０の下階層における状態Ｓ０に対するＨＤＬの記述３０３は、レジスタ動作を示すものであり、ＨＤＬの記述３０２は、このレジスタ動作に対して供給されるクロックを定義するものである。

【００４９】上記ステップＳ４０において拡張状態遷移図データが読み込まれた後は、これによって表される状態を順次調べながら、ハードウェア記述データを生成する。すなわち、まずステップＳ４２において、未調査の状態の有無を判定し、未調査の状態が有ればステップＳ４４へ進んで未調査の状態を一つ求める。このとき、同一階層に未調査の状態が存在するときは同一階層における未調査の状態を選択し、同一階層に未調整の状態が存在しないときはその上位の階層における未調査の状態を選択する。

【００５０】次のステップＳ４６では、ステップ４４で求められた状態が下階層の状態遷移を持つか（下位の状態を持つか）否かを判定し、下階層の状態遷移を持たない場合は、ステップＳ４８へ進んで、ステップＳ４４で求められた状態から出ていく矢印が有るか否かを判定する。その結果、矢印が無い場合はステップＳ４２に戻り、矢印が有る場合はステップＳ５０へ進む。

【００５１】ステップＳ５０では状態の遷移条件を求め、次のステップＳ５２において、調査対象の状態（ステップＳ４４で求められた状態）に対してレジスタ動作が記述されていれば、そのレジスタ動作を示すハードウェア記述データを生成する。その後、ステップＳ５４において状態の遷移先を求める。そしてステップＳ５６において、このようにして求められた遷移条件および遷移先（次の状態）と現在の状態とに基づき、ＦＳＭに対するハードウェア記述データを生成する。その後、ステップＳ４２に戻る。以後、未調査の状態が存在しかつステップＳ４４で求められる状態が下階層の状態遷移を持たない限り（ステップＳ４６参照）、ステップＳ４２→Ｓ４４→Ｓ４６→Ｓ４８→Ｓ５０→Ｓ５２→Ｓ５４→Ｓ５６→Ｓ４２というループ（ただし、調査対象の状態から出ていく矢印が無いときはステップＳ４８からＳ４２へ戻る）を繰り返し実行する。この実行中にステップＳ４６において下階層の状態遷移を持つと判定されると（以下、下階層の状態遷移を持つ状態を「マクロ状態」という）、ステップＳ５８へ進む。

【００５２】ステップＳ５８では、その下階層でのみ動作するクロックに対するハードウェア記述データを生成する。具体的には、まず、その下階層の上位の階層の状態遷移図における状態コードがこの状態（その下階層の状態遷移を持つマクロ状態）であることを示す値のときに「１」となり、それ以外のときに「０」となるようなフラグ信号のハードウェア記述データを生成し、次に、このフラグ信号と下階層の状態遷移用として供給すべき元のクロックとの論理積の信号を、下階層の状態コードを格納する状態レジスタへクロックとして供給することを示すハードウェア記述データ、すなわち上記フラグ信号を用いたＡＮＤゲートによるクロックのゲーティングを示すハードウェア記述データを生成する。

【００５３】上記ステップＳ５８の実行後は下階層の状態遷移に移行し、未調査の状態が存在しかつステップＳ４４で求められる状態が下階層の状態遷移を持たない限り（ステップＳ４６参照）、Ｓ４２→Ｓ４４→Ｓ４６→Ｓ４８→Ｓ５０→Ｓ５２→Ｓ５４→Ｓ５６→Ｓ４２というループ（ただし、調査対象の状態から出ていく矢印が無いときはステップＳ４８からＳ４２へ戻る）を繰り返し実行する。これにより、下階層の状態遷移図について上記と同様の処理が行われる。この実行中にステップＳ４６において更に下階層の状態遷移を持つと判定されると、その下階層について上記ループを繰り返し実行する。以降同様にして、最下位の状態遷移図に至るまで上記と同様の処理が行われる。このような処理が行われた

結果、未調査の状態が無くなれば、このサブルーチンから復帰し（ステップＳ４２）、ＨＤＬ生成処理を終了する。

【００５４】本実施形態におけるＨＤＬ生成処理では、以上のようにして、階層構造を有する拡張状態遷移図における各状態が順次調べられ、設計対象のＬＳＩに対するレジスタ転送レベルのハードウェア記述データが生成される。図２９は、このようなＨＤＬ生成処理により図２８に示した拡張状態遷移図データから得られるハードウェア記述データの一部を示す図である。この図におけるＨＤＬの記述４０２は、ステップＳ５８で生成される部分であって、下階層の状態遷移を持つマクロ状態ＨＳ０で動作していることを示すフラグＣＬＫ３ＭおよびＣＬＫ３ＭＭと、マクロ状態ＨＳ０の下階層の状態遷移用として供給すべき元のクロックＣＬＫとの論理積をとることにより、クロックのゲーティングを行ってクロックＣＬＫ３を生成することを示している。このクロックＣＬＫ３が実際に下階層の状態遷移用クロックとして供給される。また、図２９におけるＨＤＬの記述４０３もステップＳ５８で生成され、これは上記フラグＣＬＫ３ＭおよびＣＬＫ３ＭＭの作成を示している。

【００５５】本実施形態によれば、ステップ５８における上記のような処理により、各マクロ状態に対し、設計対象のＬＳＩがそのマクロ状態であるとき以外には、そのマクロ状態の下階層に対して状態遷移のクロックが供給されないような構成を記述したハードウェア記述データが生成される。したがって、このハードウェア記述データに基づいて製造されるＬＳＩでは、各マクロ状態について、そのマクロ状態の下階層の状態遷移が生じない期間中すなわちそのマクロ状態以外の状態となる期間中は、その下階層への状態遷移クロックの供給が停止し、これにより消費電力が低減される。

【００５６】上記実施形態では、下階層の状態遷移のクロックをゲーティングすることにより消費電力の低減を図っているが、下階層のレジスタ動作に対して供給されるクロックをゲーティングすることによっても消費電力を低減することができる。すなわち、まず、その下階層の上位の階層の状態遷移図における状態コードがこの状態（その下階層の状態遷移を持つマクロ状態）であることを示す値のときに「１」となり、それ以外のときに「０」となるようなフラグ信号のハードウェア記述データを生成し、次に、このフラグ信号と下階層のレジスタ動作に対して供給すべき元のクロックとの論理積の信号を下階層のレジスタ動作に対するクロックとして供給することを示すハードウェア記述データを生成するようにしてもよい。このようなハードウェア記述データに基づいて製造されるＬＳＩでは、レジスタ動作用として供給すべきクロックが上記フラグを用いてゲーティングされるため、各マクロ状態について、そのマクロ状態以外の状態となる期間中は、その下階層の状態に割り付けられたレジスタ動作に対するクロックの供給が停止する。これにより、ＬＳＩの消費電力が低減される。

【００５７】＜変形例＞以上において説明した各実施形態は、それぞれ、論理合成の効率とデバッグの容易性などを向上させるために有限状態機械ブロックとデータパス･ブロックを別の階層ブロックにするという特徴や、クロックスキューの発生防止等のために状態レジスタを複数ビット分のフリップフロップから成る一つの回路ブロックとして実現するという特徴、消費電力を低減するために下階層の状態遷移用のクロックのゲーティングを行うという特徴などを有しているが、これらの特徴を複数備えたＬＳＩ機能設計支援装置や、これらの特徴を全て備えたＬＳＩ機能設計支援装置も実現可能である。

【図面の簡単な説明】

【図１】　本発明の実施形態であるＬＳＩ機能設計支援装置のハードウェア構成を示す概略ブロック図。

【図２】　上記ＬＳＩ機能設計支援装置における処理を示すフローチャート。

【図３】　実施形態１のＬＳＩ機能設計支援装置によるＨＤＬ生成処理を示すフローチャート。

【図４】　実施形態１のＬＳＩ機能設計支援装置に入力される拡張状態遷移図データの一例を示す図。

【図５】　上記拡張状態遷移図データのデータ構造を示す図。

【図６】　従来のＬＳＩ機能設計支援装置により上記拡張状態遷移図データから生成されるハードウェア記述データを示す図。

【図７】　図６のハードウェア記述データに対応する設計対象のＬＳＩの外部ポートの入出力関係を示す図。

【図８】　実施形態１において上記拡張状態遷移図データから生成されるハードウェア記述データのうち有限状態機械ブロックの部分を示す図。

【図９】　実施形態１において上記拡張状態遷移図データから生成されるハードウェア記述データのうちデータパス･ブロックの部分を示す図。

【図１０】　実施形態１において生成されるハードウェア記述データに対応するＬＳＩの構成を示すブロック図。

【図１１】　実施形態２のＬＳＩ機能設計支援装置によるＨＤＬ生成処理におけるデータパス･ブロックのハードウェア記述の生成を示すフローチャート。

【図１２】　実施形態２において上記拡張状態遷移図データから生成されるハードウェア記述データのうち組合せ回路ブロックの部分を示す図。

【図１３】　実施形態２において上記拡張状態遷移図データから生成されるハードウェア記述データのうちレジスタ･ブロックの部分を示す図。

【図１４】　実施形態２において生成されるハードウェア記述データに対応するＬＳＩの構成のうちデータパス･ブロックの部分を示すブロック図。

19

【図１５】　実施形態３のＬＳＩ機能設計支援装置によるＨＤＬ生成処理を示すフローチャート。

【図１６】　実施形態３において生成されるハードウェア記述データに対応するＬＳＩの構成を示すブロック図。

【図１７】　実施形態４のＬＳＩ機能設計支援装置によるＨＤＬ生成処理を示すフローチャート。

【図１８】　実施形態４のＬＳＩ機能設計支援装置に入力される状態遷移図データの一例を示す図。

【図１９】　実施形態４における上記状態遷移図データの状態コード解析の結果を示す図。

【図２０】　実施形態４におけるマクロ･ライブラリに登録されているフリップフロップ･マクロを示す図。

【図２１】　実施形態４におけるハードウェア記述データ生成の際に使用されるフリップフロップ･データを示す図。

【図２２】　実施形態４のＬＳＩ機能設計支援装置に入力される状態遷移図データに対応する、状態コード割り付け後の状態遷移図を示す図。

【図２３】　従来のＬＳＩ機能設計支援装置により上記状態遷移図データから生成されるハードウェア記述データを示す図。

【図２４】　実施形態４において上記状態遷移図データから生成されるハードウェア記述データを示す図。

【図２５】　従来のＬＳＩ機能設計支援装置により生成される図２３のハードウェア記述データから論理合成により得られる論理回路を示す図。

【図２６】　実施形態４において生成される図２４のハードウェア記述データから論理合成により得られる論理＊回路を示す図。

20

＊回路を示す図。

【図２７】　実施形態５のＬＳＩ機能設計支援装置によるＨＤＬ生成処理を示すフローチャート。

【図２８】　実施形態５のＬＳＩ機能設計支援装置に入力される拡張状態遷移図データの一例を示す図。

【図２９】　実施形態５において図２８の拡張状態遷移図データから生成されるハードウェア記述データの一部を示す図。

【符号の説明】

１０　…ＬＳＩ機能設計支援装置のハードウェア本体

１２　…ハードディスク装置

１６　…ＣＰＵ

１８　…メモリ

２１１…状態コードの割付を示すハードウェア記述（実施形態４）

２１２…状態レジスタを示すハードウェア記述（実施形態４）

２１３…４ビットのフリップフロップ･マクロの使用を示すハードウェア記述（実施形態４）

４０２…状態遷移用クロックのゲーティングを示すハードウェア記述（実施形態５）

４０３…上記ゲーティングに使用されるフラグの作成を示すハードウェア記述（実施形態５）

FSM　…有限状態機械ブロック

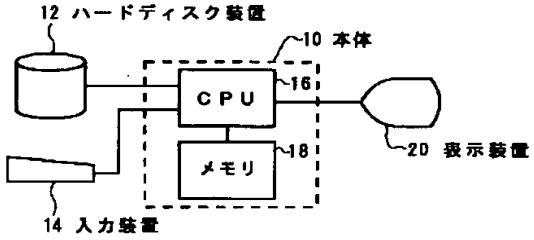
REGACT…データパス･ブロック

COMB　…組合せ回路ブロック

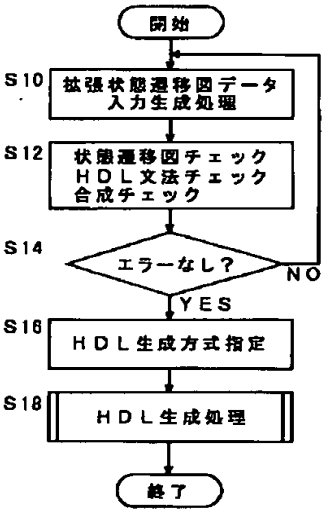
REG　…レジスタ･ブロック

R42　…４ビット･フリップフロップ･マクロ

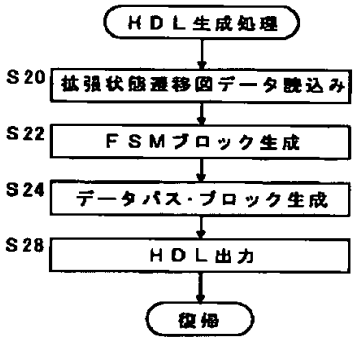
【図１】



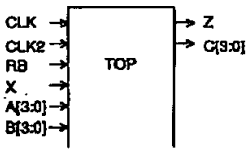
【図２】



【図３】



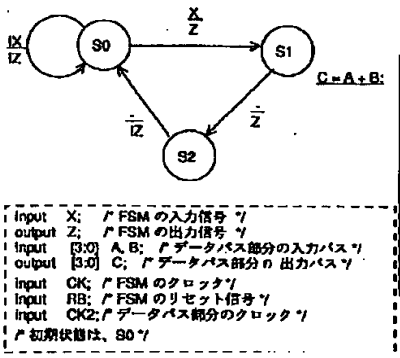
【図７】



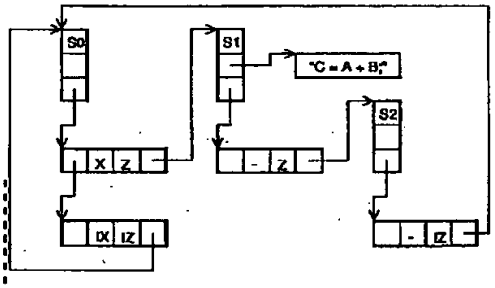
【図２０】

マクロ名	ビット数
R42	4
R82	8

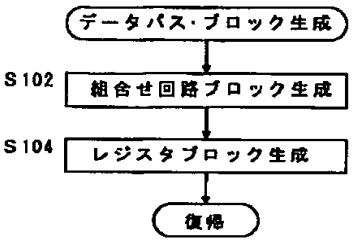
【図４】



【図５】



【図１１】



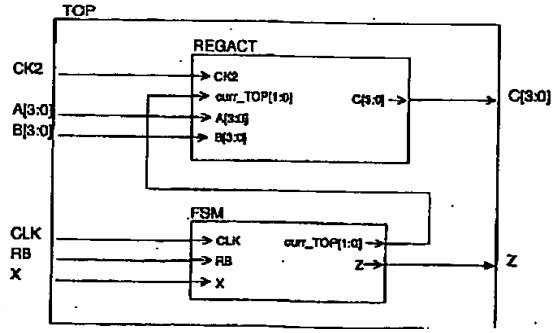
【図８】

```
module FSM ( CLK, RB, X, Z, curr_TOP);
input       CLK, RB;;
input       X;
output      Z;
reg         Z;
output      [1:0]       curr_TOP;
reg         [1:0]       curr_TOP, next_TOP;

parameter [1:0] S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

always @ ( posedge CLK or negedge RB) begin
  if ( !RB) begin
    curr_TOP <= #1 S0_TOP;
  end else begin
    curr_TOP <= #1 next_TOP;
  end
end

always @( Z or X or curr_TOP) begin
  case ( curr_TOP)
    S0_TOP : begin
      if ( X) begin
        Z = 1;
        next_TOP = S1_TOP;
      end
      else begin
        Z = 0;
        next_TOP = S0_TOP;
      end
    end
    S1_TOP : begin
      Z = 1;
      next_TOP = S2_TOP;
    end
    S2_TOP : begin
      Z = 0;
      next_TOP = S0_TOP;
    end
  endcase
end
endmodule
```

【図６】

```
module TOP ( CLK, RB, CK2, X, A, B, C, Z);
input       CLK, RB;
input       CK2;
input       X;
input       [3:0]       A, B;
output      [3:0]       C;
reg         [3:0]       C;
output      Z;
reg         Z;

parameter [1:0]
  S0_TOP = 2'b00,
  S1_TOP = 2'b01,
  S2_TOP = 2'b10;

reg [1:0]   curr_TOP, next_TOP;

always @( posedge CK2) begin
  case ( curr_TOP)
    S0_TOP : begin
      nop;
    end
    S1_TOP : begin
      C = A + B;
    end
    S2_TOP : begin
      nop;
    end
  endcase
end

task nop;
  begin
  end
endtask     /* nop */

always @ ( posedge CLK or negedge RB) begin
  if ( !RB) begin
    curr_TOP <= #1 S0_TOP;
  end else begin
    curr_TOP <= #1 next_TOP;
  end
end

always @( Z or CK2 or X or A or B
          or C or curr_TOP) begin
  case ( curr_TOP)
    S0_TOP : begin
      if ( X) begin
        Z = 1;
        next_TOP = S1_TOP;
      end
      else begin
        Z = 0;
        next_TOP = S0_TOP;
      end
    end
    S1_TOP : begin
      Z = 1;
      next_TOP = S2_TOP;
    end
    S2_TOP : begin
      Z = 0;
      next_TOP = S0_TOP;
    end
  endcase
end
endmodule
```

【図１０】



【図１３】

```
module REG ( CK2, next_C, C);
input               CK2;
input       [3:0]   next_C;
output      [3:0]   C;
reg         [3:0]   C;

always @( posedge CK2) begin
  C = next_C;
end

endmodule
```

【図１８】

状態名	次の状態	出力
S0	S1	
S1	S2	Z
S2	S3	Z
S3	S4	
S4	S5	
S5	S6	
S6	S7	Z
S7	S8	Z
S8	S0	

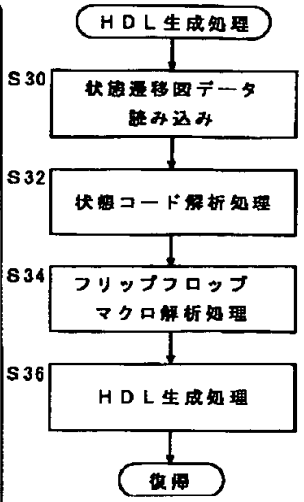
【図９】

```
module REGACT ( CK2, curr_TOP, A, B, C);
input       [1:0]       curr_TOP;
input       [3:0]       A, B;
output      [3:0]       C;
reg         [3:0]       C;

parameter [1:0] S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

always @( posedge CK2) begin
   case ( curr_TOP)
      S0_TOP : begin
         nop;
      end
      S1_TOP : begin
         C = A + B;
      end
      S2_TOP : begin
         nop;
      end
   endcase
end

task nop;
   begin end
endtask      /* nop */

endmodule
```

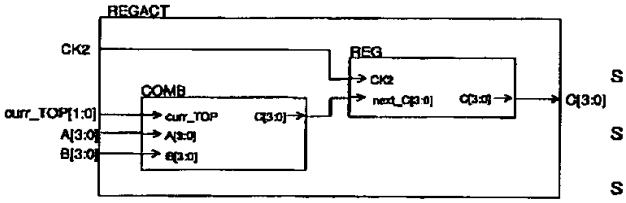
【図１２】

```
module COMB ( curr_TOP, A, B, C);
input       [1:0]       curr_TOP;
input       [3:0]       A, B;
output      [3:0]       C;
reg         [3:0]       C;

parameter [1:0] S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

always @( A or B) begin
   case ( curr_TOP)
      S0_TOP : begin
         nop;
      end
      S1_TOP : begin
         C = A + B;
      end
      S2_TOP : begin
         nop;
      end
   endcase
end

task nop;
   begin end
endtask      /* nop */

endmodule
```

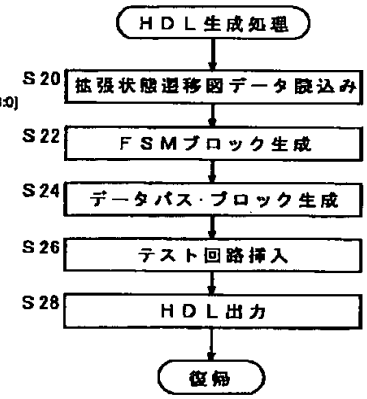
【図１７】



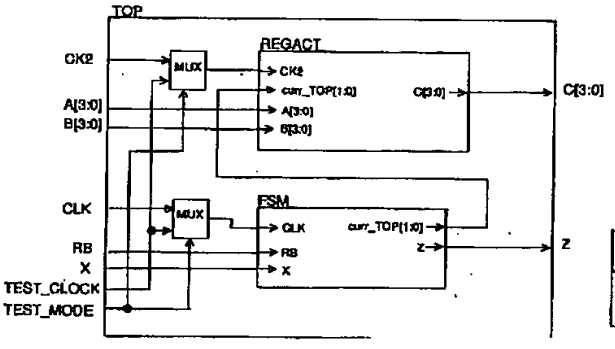
【図１４】



【図１５】



【図１６】



【図１９】

(a)

状態数	ビット数
9	4

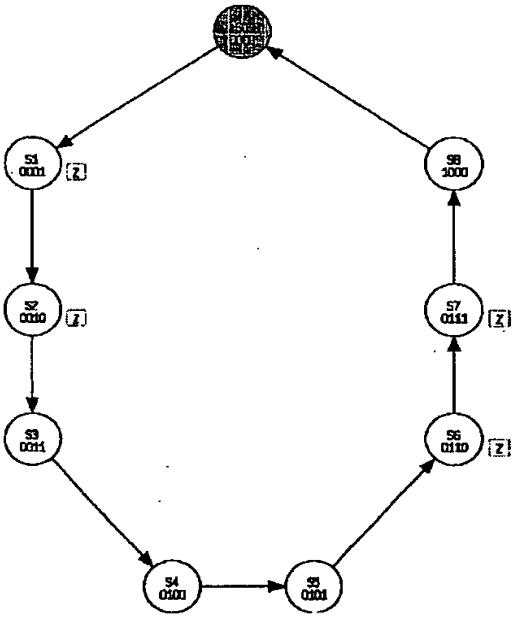
(b)

状態名	状態コード
S0	0000
S1	0001
S2	0010
S3	0011
S4	0100
S5	0101
S6	0110
S7	0111
S8	1000

【図２１】

マクロ名	ビット数
R42	4

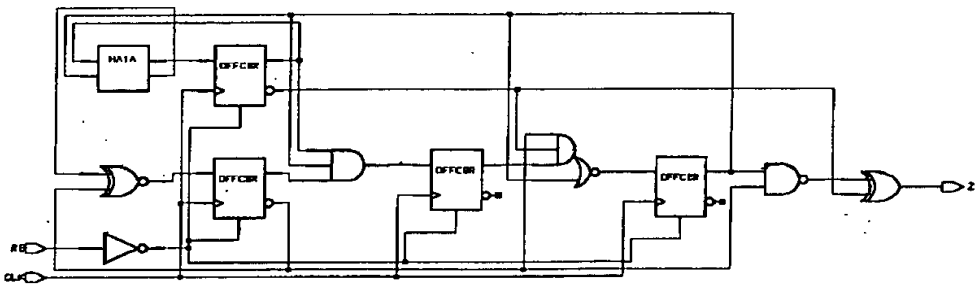
【図２２】



【図２３】

```
module SAMPLE ( CLK, RB, Z);
input   CLK, RB;
output  Z;
reg     Z;

parameter [3:0] // synopsys enum state_State
    S0_State = 4'b0000,
    S1_State = 4'b0001,
    S2_State = 4'b0010,
    S3_State = 4'b0011,
    S6_State = 4'b0110,
    S7_State = 4'b0111,
    S8_State = 4'b1000,
    S4_State = 4'b0100,
    S5_State = 4'b0101;                                          ~201

reg [3:0]       /* synopsys enum state_State */ State, next_State;  <-- 202
// synopsys state_vector State


always @ ( posedge CLK or negedge RB) begin
    if ( !RB) begin
        State <= #1 S0_State;
    end else begin
        State <= #1 next_State;
    end
end

always @( Z or State) begin
    case ( State)  //synopsys parallel_case full_case
        S0_State : begin
            Z = 0;
            next_State = S1_State;
        end
        S1_State : begin
            Z = 1;
            next_State = S2_State;
        end
                                  :
                                  :
                                  :
        S5_State : begin
            Z = 0;
            next_State = S6_State;
        end
    endcase
end

endmodule
```

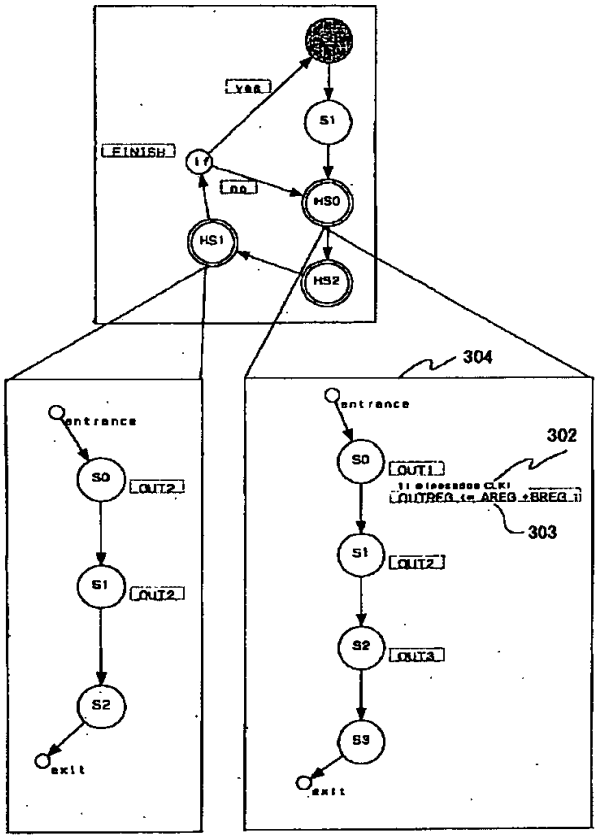
【図２５】



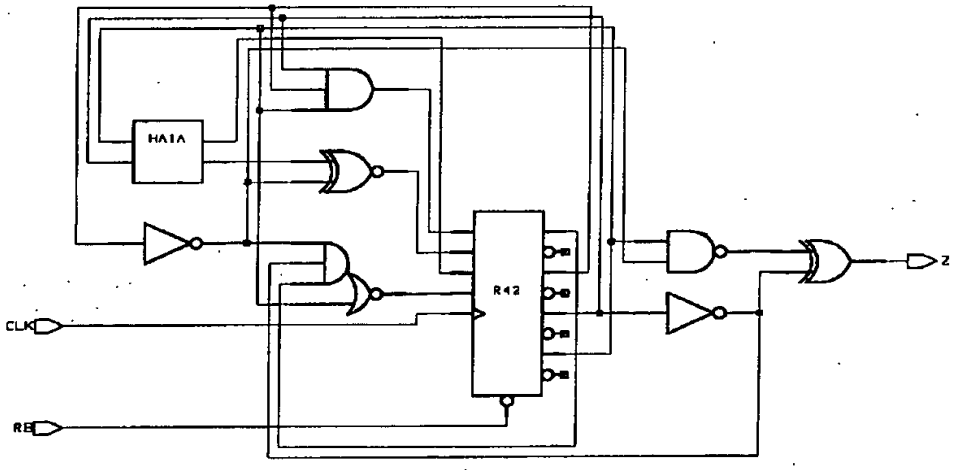
【図２４】

```
module SAMPLE_MULTI ( CLK, RB, Z);
input   CLK, RB;
output  Z;
reg     Z;

parameter [3:0] // synopsys enum state_State
    S0_State = 4'b0000,
    S1_State = 4'b0001,
    S2_State = 4'b0010,
    S3_State = 4'b0011,
    S6_State = 4'b0110,
    S7_State = 4'b0111,
    S8_State = 4'b1000,
    S4_State = 4'b0100,
    S5_State = 4'b0101;                                   ~211

reg  [3:0] next_State;
wire [3:0] State;  <---- 212                              ~213

/* synopsys dc_script_begin
   set_dont_touch find ( reference, ( R42 ) )
*/
R42 r42 (.QA(State[3]),.QB(State[2]),.QC(State[1]),.QD(State[0]),.CK(CLK),.RB(RB),
         .A(next_State[3]),.B(next_State[2]),.C(next_State[1]),.D(next_State[0]));


always @( Z or State) begin
    case ( State)  //synopsys parallel_case full_case
        S0_State : begin
            Z = 0;
            next_State = S1_State;
        end
        S1_State : begin
            Z = 1;
            next_State = S2_State;
        end

        S5_State : begin
            Z = 0;
            next_State = S6_State;
        end
    endcase
end

endmodule
```

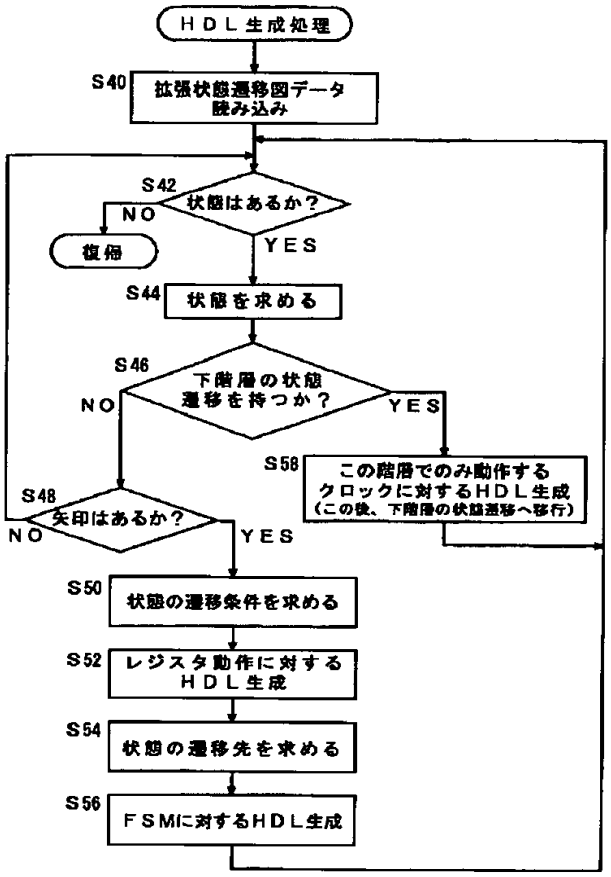
【図２８】



【図２６】



【図２７】



【図２９】

```
module TESTLP ( RB, FINISH, CLK, AREG, BREG, OUTREG, OUT1, OUT2, OUT3);
input   RB;
input FINISH, CLK ;
input [7:0] AREG,BREG;
output [7:0] OUTREG;
reg    [7:0] OUTREG;

output  OUT1, OUT2, OUT3;
reg     OUT3;
reg     OUT1_HS0, OUT1_HS2, OUT2_HS0, OUT2_HS2, OUT2_HS1;

reg     ins_HS0_test, ins_HS2_test, ins_HS1_test, ex_HS0, ex_HS2; ex_HS1;
                                  :
wire CLK3 ;
assign CLK3 = CLK & (CLK3N|CLK3MM) ;                    ~402

wire CLK4 ;
assign CLK4 = CLK & (CLK4N|CLK4MM) ;

wire CLK5 ;
assign CLK5 = CLK & (CLK5N|CLK5MM) ;

always @ ( negedge CLK or negedge RB ) begin
    if( !RB )begin
       CLK3N <= 0 ;                                     ~403
    end else begin
       CLK3N <= !ex_HS0 | ins_HS0_test ;
    end
end

always @ ( negedge CLK or negedge RB ) begin
    if( !RB ) begin
       CLK3MM <= 0 ;
    end else begin
       CLK3MM <= CLK3N ;
    end
end

always @ ( posedge CLK3 or negedge RB) begin
    if ( !RB) begin
        curr_HS0 <= #1 idle_HS0;
    end else begin
        curr_HS0 <= #1 next_HS0;
    end
end
```